

Multi-robot coverage to locate fixed targets using formation structures

J.A. Rogge^a, D. Aeyels^a

^a*SYSTeMS Research Group, Ghent University, Technologiepark Zwijnaarde 914, 9052 Zwijnaarde, Belgium*

Abstract

This paper develops an algorithm that guides a multi-robot system in an unknown environment in search of fixed targets. The area to be scanned contains an unknown number of convex obstacles of unknown size and shape. The algorithm covers the entire free space in a sweeping fashion and as such relies on the use of robot formations. The geometry of the robot group is a lateral line formation, which is allowed to split and rejoin when passing obstacles. It is our main goal to exploit this formation structure in order to reduce robot resources to a minimum. Each robot has a limited and finite amount of memory available. No information of the topography is recorded. Communication between two robots is only possible up to a maximum inter-robot distance, and if the line-of-sight between both robots is not obstructed. Broadcasting capabilities and indirect communication are not allowed. Supervisory control is prohibited. The number of robots equipped with GPS is kept as small as possible. Applications of the algorithm are mine field clearance, search-and-rescue missions, and intercept missions. Simulations are included and made available on the internet, demonstrating the flexibility of the algorithm.

Keywords: multi-robot systems, coverage, decentralized control, exploration.

1. Introduction

1.1. Multi-robot coverage based on single-robot algorithms

The research domain of multi-agent mobile robot systems consists of subdomains according to the task to be performed by the robot group [1]. At present, well-studied subdomains are motion-planning (also called path-planning), formation-forming, region-sweeping, and combinations thereof. This paper discusses region-sweeping. Region-sweeping algorithms appear in two distinct types of robot missions: mapping of an area and coverage of an area. A mapping algorithm produces a topographical map of the area; a coverage algorithm is performed successfully when all free space has been covered by the robots, without demanding a map. We restrict our attention to coverage problems where teams of robots are involved.

We briefly recall the most relevant approaches to multi-robot coverage missions. The most basic approach lets the robots move around independently in a random fashion [2, 3]. With time tending to infinity the entire area gets covered.

A more structured approach extends existent exploration algorithms for a single robot to the multi-robot case with unlimited communication capabilities. Unlimited communication can be realized in two ways:

1. every robot is able to broadcast messages to all other robots irrespective of inter-robot distance,

2. every robot can exchange information with a shared memory unit.

A typical strategy combining single-robot coverage techniques, consists of dividing the exploration area into separate regions, each of which is assigned to a single robot [4]. Efficient ways to assign these regions to the robots are discussed in [5, 6, 7, 8]. These algorithms use a supervisor for the assignment and require a-priori knowledge on the lay-out of the environment. Throughout the algorithm the robots transmit information to each other (or a central shared memory unit) with regard to the area they have covered so far, in order to minimize the probability of covering the same area more than once, reducing the operating time of the algorithm significantly.

Other research groups have developed robot systems that combine single-robot coverage techniques as described above, but assume more realistic situations with one or more of the following assumptions:

- The communication and sensing properties of the robots are limited,
- the environment is unknown,
- no supervisory control or shared memory is allowed.

In [9] for instance, a strategy is used that does not require a supervisor to distribute the area between robots: each robot is guided by a neural network representing the (known) environment. This network ensures the robot is globally attracted towards unscanned areas. Cooperation among robots consists of collision avoidance between them.

Email addresses: jonathan.rogge@ugent.be (J.A. Rogge), dirk.aeyels@ugent.be (D. Aeyels)

In other words communication is limited to robots with sufficiently small inter-robot distance.

A line of research considering robot coverage where all three of the above assumptions hold, consists of so-called ant-robotics [10, 11, 12]. Ant-robots use very limited communication and need hardly any memory. Their behavior is inspired by ant behavior as found in nature: the robots are able to leave so-called pheromone traces in the environment. This pheromone serves as a means of indirect communication replacing inter-robot communication and a shared memory. The environment is divided into a grid; pheromone is represented by a quantity that assumes a value in every cell of the grid and determines the behavior of the robot in the respective cell. In essence, the coverage strategy with ant-robotics is identical to the strategy described before: each robot in the team performs single-robot coverage. The pheromone indicates which areas have already been covered, preventing the same area to be covered twice. In experimental setups pheromone is represented by a chemical substance, a heat trail, pen marks, or crumbs. The choice needs to be adjusted to the complexity of the algorithm under consideration. Implementation is not always straightforward and is the subject of on-going research [13].

1.2. Multi-robot coverage based on robot formations

There exists an approach to the coverage problem which does not resort to transposing single-robot algorithms to a multi-robot setting, but takes advantage of the cooperation capabilities of a robot team. The robot team or its subgroups maintain formations to scan the area [14, 15, 16, 17]. The degree of rigidity of and cooperation inside the formations may vary depending on the approach. The environment is represented by standard Euclidean space instead of a static grid structure. A successful covering approach with robot formations uses the so-called boustrophedon decomposition [17]. This is a cellular decomposition of the unknown environment, where each cell has the property that it can be scanned by back-and-forth motions of the robot team. The robot team itself creates this decomposition in an online fashion, i.e. during the algorithm's execution. Although this approach is able to tackle an unknown environment without using a supervisory control, a major draw-back remains the heavy use of sensing, communication and memory capabilities of the robots. Every robot needs to store the cellular decomposition of the environment into its memory. It does this in the abstracted form of a so-called Reeb graph. The topology of the Reeb graph depends on the shape and position of the obstacles. Vertices of the graph correspond with specific corners of the obstacles. The robots need to record the coordinates of these points using GPS and store them into their memory for later use. The online construction of a Reeb graph is not straightforward, as explained in [18]. The robots exchange information on the graph in order to obtain a global picture of the environment, which enables them to keep track of areas left to be covered. Moving to

uncovered areas may happen over long distances through already covered terrain.

1.3. Our approach

The present paper combines ideas from ant-robotics and coverage with robot formations, leading to a novel approach of the coverage problem. In line with the philosophy of ant-robotics where simplicity of the robots is a key element (recall the three assumptions on communication/sensing, the environment and supervisory control mentioned above), this paper investigates the possibility of significantly reducing the necessary resources of the robots. Instead of resorting to indirect communication through pheromone, we investigate the benefit brought about by robot formations to guarantee coverage. The robots' resources are limited in the following sense. The sensing capabilities of each robot are of limited range. Inter-robot communication is allowed for robots which are located sufficiently close to one another. Communication with a shared memory unit or via pheromone is prohibited. Decision-making is fully decentralized: each robot determines its actions based on its own observations and memory content. The lay-out of the environment to be covered is not known a-priori. Moreover, to reduce the demands on each robot's memory, a map or abstract representation of the explored environment is not stored in any kind of memory device. The memory use of each robot is reduced and does not scale with the size of the problem or the size of the robot team. Finally, as few robots as possible know their absolute position. To the best of our knowledge, no other approach in the literature is able to reduce the robots' resources to the same degree. However, putting all these limitations in place comes at a cost: the algorithm constructed in the present paper guarantees full coverage of spaces containing only convex obstacles. If we want to be able to treat (subclasses of) nonconvex obstacles, we need to relax the aforementioned limitations.

Our coverage method, a preliminary version of which has appeared in [19], does not apply a dynamic cellular decomposition as described in Section 1.2. We let the robot group sweep the area in one or more predefined *scanning strips*. These strips do not take the presence of obstacles into account: obstacles are allowed to intersect the boundaries of adjacent strips. The robot group sweeps the area strip by strip in a back-and-forth fashion while dealing with the obstacles encountered inside the strip. The size of the strips depends on the number of robots. The location of the strips is communicated to the outer robots of the formation at the onset of the algorithm. These two robots follow the strip boundaries using a GPS system. The remaining robots stay in formation which leads to a successful coverage of a strip. The algorithm is described in detail in Section 5. We also provide a proof that the algorithm guarantees full coverage (see Section 6).

Some applications we have in mind are mine field clearance using chemical vapor microsensors (introduced in [20]) and search-and-rescue of snow avalanche victims, using

specialized transceivers. In both cases the robots are equipped with specialized sensors to locate the targets. When a target is detected a signal is given to start negotiating the target. In case of mine field clearance this would be dismantling the mine, in search and rescue this is evacuating the victim.

2. Defining the setting

2.1. Modeling the environment

We single out a rectangular area $S \subset \mathbb{R}^2$ that we want to explore. The set S is divided into several parallel rectangles with equal width w . These rectangles are called scanning strips. We define a right-handed (x, y) -frame, with the y -axis directed along the common boundaries of the scanning strips (see Figure 1).

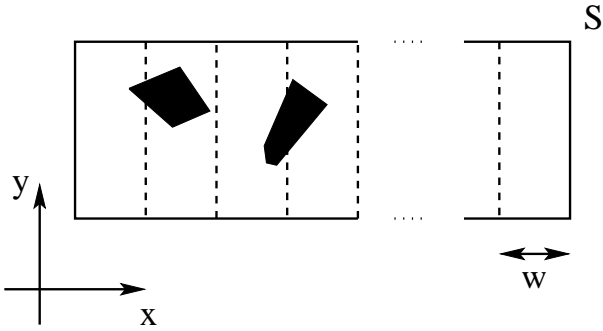


Figure 1: The area S assigned to the scanning algorithm; the black polygons represent obstacles.

Obstacles inside S are represented by polygons. With N_O denoting the number of obstacles in S , each obstacle obtains an index $i_O \in \mathcal{N}_O := \{1, \dots, N_O\}$.

The maximal diameter γ of the obstacles is defined by

$$\gamma := \max_{i_O} \max_{p, q \in P_{i_O}} \|p - q\|, \quad (1)$$

with P_{i_O} an obstacle in S . Furthermore it is assumed that the distance between obstacles is sufficiently large. Further on in the paper a lower bound on the inter-obstacle distance is given (see Section 5.3.2).

Related to the geometrical description of the obstacles is the concept of *top of an obstacle*, which plays an important role in the construction of the coverage algorithm and its analysis.

Definition 1. A top $p = (p_x, p_y) \in \mathbb{R}^2$ of a (convex) obstacle P is a point satisfying the following property:

$$\exists \delta > 0, \forall \epsilon < \delta : q = (q_x, q_y) \in N(p, \epsilon) \cap P \Rightarrow q_y \leq p_y,$$

where

$$N(p, \epsilon) := \{z \in \mathbb{R}^2 \mid \|z - p\| < \epsilon\}.$$

The “top” is defined relative to the direction of motion of the robot group: Definition 1 assumes a robot group moving parallel to the y -axis towards larger y -values. If the robot group moves in the opposite direction, the definition changes correspondingly by demanding $q_y \geq p_y$. A convex obstacle either has exactly one top or possesses a connected line of tops.

2.2. Robot sensors and communication

Each robot is equipped with two types of omnidirectional sensors. One type serves as a means to detect the targets in the assigned area, e.g. landmines. Its detection range is denoted r_t^+ . The other type is responsible for detecting obstacles and other robots, with detection range r_r^+ . The simplest sensor model available is the binary detection model: a target is detected (not detected) with complete certainty if it is in (outside) the sensor’s detection range [21]. More realistic descriptions of a sensor’s detection capability use probabilistic models [22]. A function $c_p : \mathbb{R}^2 \rightarrow [0, 1]$ expresses the *coverage confidence level*: the value $c_p(q)$ represents the probability that the sensor located at p detects an object that is located at q . We assume the confidence level only depends on the distance between p and q , which leads to a circular symmetry around the sensor. We further assume the confidence level to be defined as follows:

$$c_p(q) = \begin{cases} 1, & \|p - q\| \leq r^-, \\ \frac{e^{-\|p - q\|} - e^{-r^+}}{e^{-r^-} - e^{-r^+}}, & r^- < \|p - q\| < r^+, \\ 0, & \|p - q\| \geq r^+. \end{cases} \quad (2)$$

In a circular area around the sensor (with radius $r^- \in \mathbb{R}_{>0}$) targets are always detected. this area is surrounded by a ring-shaped area (with radius inside (r^-, r^+)) where targets are detected with a probability smaller than 1. Outside the area with radius r^+ targets are never detected. We use (2) to describe the sensor capabilities, by adding the subscript r for sensors detecting other robots or obstacles and t for sensors detecting the targets specified in the robot coverage mission.

Furthermore, each robot possesses a compass enabling the robot to determine its orientation within space. The compass is used to align all robots along the same absolute reference direction, parallel to the strip boundary. The two outer robots of the group (robots 1 and 6 in Figure 3) are equipped with a GPS system to determine their position as explained in the introduction.

Communication is limited: first, it is only allowed between robots that sense each other. Two robots sense each other if and only if they are sufficiently close to each other (expressed by the maximum detection range r_r^+) and if there is no obstacle located on the straight line connecting them (so-called line-of-sight communication). Second, each robot transmits a *limited* set of messages, concerning its status, with the purpose of inducing a change of behavior in other robots.

2.3. Robot formation

Consider a population of N robots. To every robot one assigns an index number $i \in \{1, \dots, N\}$, which serves as the robot's identity. Each robot i with $2 \leq i \leq N$ has an *Immediate Leader* (denoted IL) with index $i - 1$. Similarly each robot i with $1 \leq i \leq N - 1$ has an *Immediate Follower* (IF), with index $i + 1$. The position of robot i is given by $q_i := (x_i, y_i) \in S$. We assume holonomic robots with discrete dynamics

$$q_i[k+1] = q_i[k] + u_i[k], \quad 1 \leq i \leq N, \quad k \in \mathbb{N}, \quad (3)$$

where $u_i \in \mathbb{R}^2$ is the control input to the i th robot. The input is bounded: $\|u_i(k)\| \leq \tilde{v}, \forall i, k$, where \tilde{v} is the distance traveled during one unit of time when moving at the maximum allowed velocity v_{\max} .

The robot formation used throughout the algorithm is defined by relative positions among the robots. We introduce the following definition (depicted in Figure 2).

Definition 2. With $d \in \mathbb{R}_{>0}$, the *Left Neighbor Position* (LNP) of a robot at (x, y) is the point with coordinates $(x - d, y)$; similarly the *Right Neighbor Position* (RNP) is the point $(x + d, y)$.

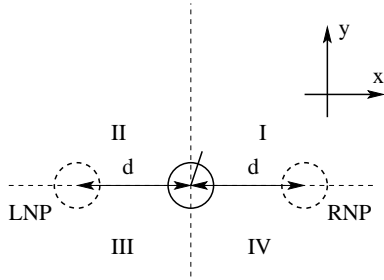


Figure 2: Schematic representation of the scanning of consecutive scanning strips.

The robot formation is obtained when the IL of each robot is located at this robot's LNP. This yields a straight-line robot formation oriented perpendicular to the direction of motion along the y -axis, as shown in Figure 3 for $N = 6$. The solid circles delimit the area sensed for targets by each robot with coverage confidence level equal to one; they have radius r_t^- . These circles overlap if the inter-robot distance d satisfies $d \leq 2r_t^-$. We set $d = 2r_t^-$ such that along the line connecting two neighboring robots in formation, targets are detected with certainty. Similarly, it is demanded that neighboring robots detect each other with a probability equal to one. Therefore we require $d \leq r_r^-$. In Figure 3 this corresponds with overlapping dashed circles with a robot located in each intersection.

Remark. Notice that Figure 3 indicates inter-robot distances as the distance between the robots' centers. This is the definition of inter-robot distance used throughout the paper. It allows us to neglect the real size of the robots in a theoretical approach of the scanning algorithm. In practice

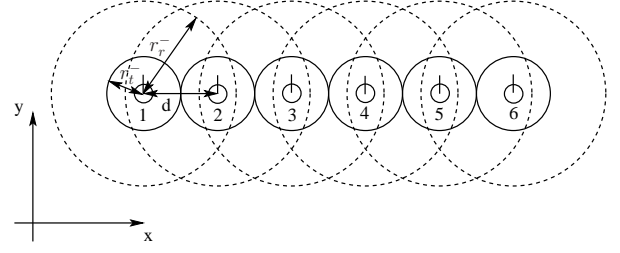


Figure 3: A desired robot configuration of 6 robots. The small solid circles represent the robots; the large solid circles show the area where targets are detected with probability equal to one; the dashed circles indicate the area where robots are able to detect each other with probability equal to one.

however, the distance sensors of the robot are attached to the robot's outer frame. A version of the algorithm used in practice needs appropriate adjustments to take the robot size into account.

2.4. Problem statement

Call \mathcal{P} the set of all points of S belonging to obstacles: $\mathcal{P} := \{q | q \in P_{i_O}, i_O \in \mathcal{N}_O\}$, with P_{i_O} an obstacle. In order to locate all fixed targets, we need the robots to cover $S \setminus \mathcal{P}$. We demand the robots cooperate with each other by maintaining the formation defined in Section 2.3. The robot formation performs a sweep of each scanning strip. The consecutive strips are scanned in opposite directions as illustrated in Figure 4. The challenge is to pass all obstacles, with unknown size and location, so that coverage is still guaranteed. The robot group is allowed to split into subgroups to move past obstacles. The shape of the robot formation, i.e. the straight line, will allow for easy and accurate reconnection of subgroups.

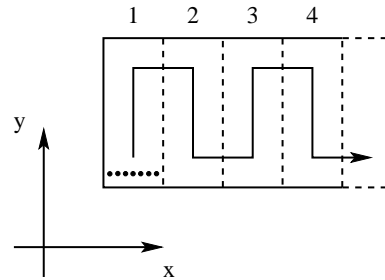


Figure 4: Schematic representation of the scanning of consecutive scanning strips.

Remark. The algorithm alternates between stages where the robot group advances towards larger y -values and stages where it moves towards lower y -values. In Section 5, the algorithm is explained and described for the former stages, where the above Definition 2 for LNP and RNP is valid. In the latter stages, where the robot group has reversed

its traveling direction, the LNP and RNP can be redefined by simply substituting $-d$ for d .

3. Preliminaries to the scanning algorithm

3.1. Preliminary definitions

Definition 3. A location in space is called covered at time t_1 if it was found inside the sensor range r_t^- of one of the robots at some time instant $t \leq t_1$.

Definition 4. A location in space is called obstructed to a robot if the straight line connecting that location with the present position of the robot intersects an obstacle or other robot.

Definition 5. A location in space is called reachable to a robot if it is not obstructed and the robot is able to reach it at the preset velocity v in one time step.

Definition 6. The robots are classified into three groups according to their location in the formation. The leftmost and rightmost robots are called the Left Strip Boundary robot, denoted LSB, and the Right Strip Boundary robot (RSB). The third group consists of all remaining robots; they are called Interior Robots (IR).

With the (x, y) -frame defined in Section 2.1 we are able to define positions w.r.t. obstacles. With y_{min} and y_{max} defined as the smallest resp. largest y -value of an obstacle P we define

1. “on the left of P ” := located at one or more points (x_i, y_i) with $y_i \in (y_{min}, y_{max})$ and $x_i \notin P$ such that $\exists x \in P: x_i < x$,
2. “on the right of P ” := located at one or more points (x_i, y_i) with $y_i \in (y_{min}, y_{max})$ and $x_i \notin P$ such that $\exists x \in P: x_i > x$.

3.2. Necessary sensor data

Since every robot is equipped with a compass it is able to retrieve the orientation of the (x, y) -frame defined at initialization. Each robot divides its surroundings into *four quadrants* with respect to the frame that

- is a translated version of the initial (x, y) -frame and
- has the center of the robot located at the origin.

These quadrants are denoted by Roman numerals in Figure 2. Furthermore, every robot is equipped with sensors measuring both

- the distance between itself and nearby obstacles and robots,
- the direction along which these obstacles and robots are detected. The straight half-line along this direction belongs to one of the robot’s four quadrants.

In the case of sensed obstacles, the robot stores into its memory

- the shortest distance between itself and the surrounding obstacles, denoted Dist2O (shorthand for “Distance w.r.t. Obstacles”),
- the quadrant corresponding to this shortest distance.

In the case of sensed robots, the robot stores into its memory all inter-robot distances and corresponding directions.

3.3. A robot’s parameters

Besides its position in S , the state of every robot is determined by a number of parameters.

3.3.1. Behavior type (TYPE)

The TYPE parameter assumes one of three values: Run-Mode, Contour-Following, and Standstill:

- Standstill: do not move.
- Contour-Following: move clockwise along the boundary of the nearest obstacle.
- Run-Mode: If the IL of a robot i is not within sensor range or it is Contour-Following, robot i moves forward with preset velocity v ; otherwise, i.e. if the IL is located at (x_i, y_i) within sensor range and is not Contour-Following, robot i moves to $(x_i + d, y_i)$, called the Desired Position (DP).

3.3.2. “Located at the top of an obstacle (TOP)”

In some instances it is necessary to mark the robot as being located at the top of an obstacle. The parameter values of TOP are “on” and “off”.

3.3.3. “Located left of an obstacle (LEFT)”

This parameter indicates when a Standstill robot (group) is located on the left of the nearest obstacle. The parameter values of LEFT are “on” and “off”.

3.3.4. “Located right of an obstacle (RIGHT)”

This parameter indicates when a Standstill robot (group) is located on the right of the nearest obstacle. The parameter values of RIGHT are “on” and “off”.

4. Scanning one strip without obstacles

Consider a scanning strip devoid of obstacles. The width of the scanning strip w and the number of robots in the formation N are interdependent. From the settings in Section 2.3, the formation width equals $(N - 1)d$. Referring to (2), the distance between each outer robot and its respective strip boundary is set to r_t^- , to ensure detection of targets located near the strip boundary. This yields a corresponding strip width

$$w = (N - 1)d + 2r_t^-. \quad (4)$$

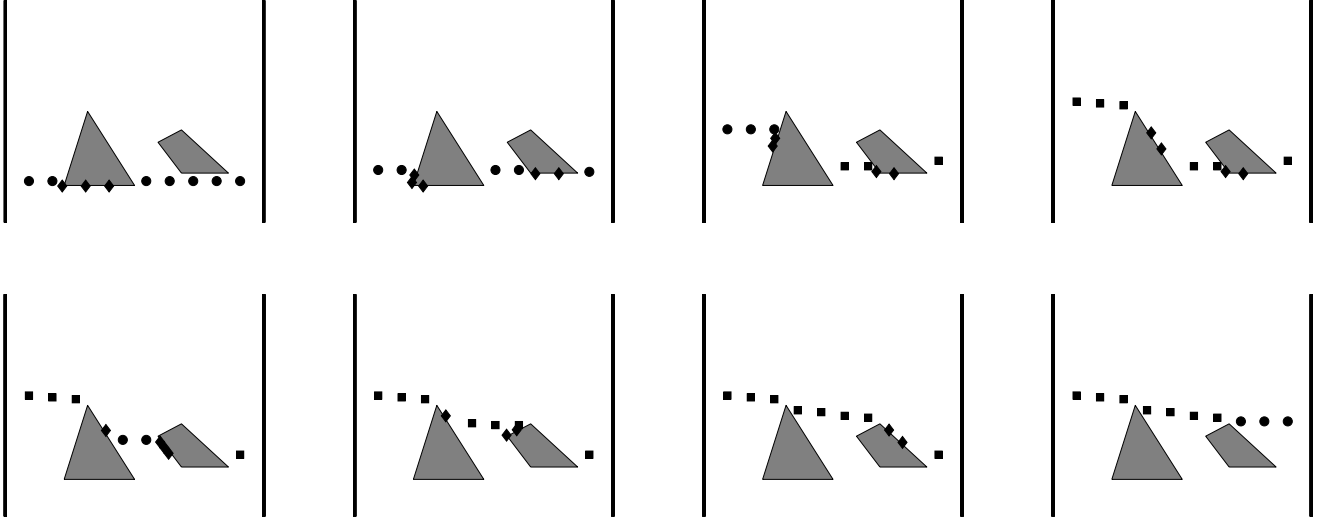


Figure 6: A 10-robot group passing two disjoint obstacles simultaneously. Squares = robots at Standstill; diamonds = Contour-Following robots; disks = robots in Run-Mode.

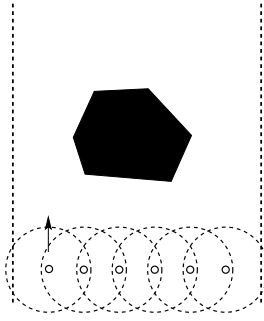


Figure 5: A depiction of the algorithm.

Given a value for w , (4) yields the necessary number of robots to scan the strip in one sweep.

At the initialization of the algorithm, the robot group has assumed the formation defined in Section 2.3. With (x_1, y_1) the coordinates of LSB, the horizontal position of the i th robot is $x_i = x_1 + (i - 1)d$. The horizontal line $y = y_1$ is covered by the robot sensors for all $x \in [x_1 - r_t^-, x_N + r_t^-]$. Robot LSB executes its algorithm which consists of tracing a straight line $x = x_1, y \geq y_1$ at a constant velocity v . The remaining robots maintain the formation and move along their respective straight line $x = x_i, y \geq y_1$. When the LSB stops at (x_1, y_2) the corresponding area $A := [x_1 - r_t^-, x_N + r_t^-] \times [y_1, y_2]$ has been completely covered by the sensors. In other words, if the robots track the parallel lines with x-coordinates $x_1 + id, i = 1, \dots, N$, the entire area gets covered.

When the robot group reaches the end of a strip, it moves to the start of the next strip. All robots turn 180 degrees, the LSB and RSB exchange roles, and the robot team commences a new sweep.

5. Scanning one strip with obstacles

Before presenting a detailed description of our algorithm, we give a brief sketch of the main idea. Every robot of the formation moves along its predefined straight line, until the line is obstructed by an obstacle, in which case the robot moves clockwise around the obstacle. Subgroups of robots which are not hindered by the obstacle keep advancing past obstacles up to a maximum distance from the obstacle. These subgroups then wait for robots moving along the obstacle to (re)join, after which the enlarged subgroup is able to advance further. In this way subgroups build up alongside the obstacle until eventually two subgroups from each side of the obstacle join at the obstacle's top. This idea is depicted in Figure 6, where two obstacles are being passed simultaneously. The figure is the result of a computer simulation implementing the algorithm as it is described below.

In more detail, the solution to the problem statement of Section 2.4 consists of endowing each robot with the behavior types Standstill, Run-Mode, and Contour-Following defined in Section 3.3. These behavior types are each triggered by obstacles and/or other robots. The algorithm itself consists of a set of rules that determine the conditions forcing appropriate transitions between the three behavior types. These rules are described in 3 tables, one for each behavior type (see Algorithm 1, Algorithm 2, and Algorithm 3). As will be explained in the present section, they lead to successful coverage if the minimum inter-obstacle distance is given by $2\sqrt{2}d$.

5.1. Switching from Run-Mode to Contour-Following

The robot group is initialized with all robots in Run-Mode and all parameters TOP, LEFT, and RIGHT turned OFF. The group advances inside the strip in formation

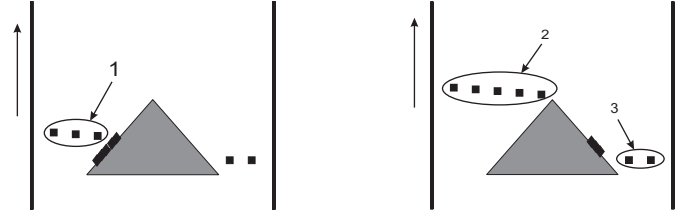


Figure 7: Three typical situations where a subgroup of Standstill robots appears. Square = Standstill robot; diamond = Contour-Following robot.

Algorithm 1 TYPE Run-Mode

```

Repeat
  if IL is visible  $\wedge$  its TYPE is not Contour-Following
  then
    keep your IL at your LNP
  else
5:   Advance with constant  $x$ -coordinate
  end if
until
  if An obstacle is blocking the forward path then
    TYPE  $\leftarrow$  Contour-Following
10:  else if an obstacle in quadrant 4  $\wedge$  dist2O  $> d \wedge$  your
    IF is not at your RNP then
      LEFT  $\leftarrow$  ON
      TYPE  $\leftarrow$  Standstill
    else if your IF is at your RNP, with
    TYPE(IF)=Standstill and LEFT(IF)=ON then
      LEFT  $\leftarrow$  ON
15:   TYPE  $\leftarrow$  Standstill
    else if an obstacle in quadrant 3  $\wedge$  dist2O  $> d \wedge$  your
    IL is not at your LNP then
      RIGHT  $\leftarrow$  ON
      TYPE  $\leftarrow$  Standstill
    else if your IL is at your LNP, with
    TYPE(IL)=Standstill and RIGHT(IL)=ON then
20:   RIGHT  $\leftarrow$  ON
      TYPE  $\leftarrow$  Standstill
  end if

```

until it encounters an obstacle. In the presence of obstacles, a robot switches **from Run-Mode to Contour-Following**

- if its Desired Position is located inside an obstacle and hence is unreachable, or,
- if its IL is Contour-Following and the forward path is obstructed by an obstacle.

5.2. Switching from Run-Mode to Standstill

If the above conditions are not satisfied, the robot remains in Run-Mode and moves past the obstacle. Consider a robot which passes on the left of an obstacle and has lost its IF at its RNP because that robot started Contour-following around the obstacle. This can only happen if the distance between the robot and the obstacle decreased to a value less than d . While the robot advances, its distance with the obstacle will eventually increase again. The robot switches **from Run-Mode to Standstill** if Dist2O $> d$ and the corresponding obstacle is in the robot's fourth quadrant. The robot's LEFT is turned ON. Similarly, if a Run-Mode robot (apart from LSB) has no IL at its LNP and if Dist2O $> d$ and the corresponding obstacle is in the robot's third quadrant, it is forced to switch to Standstill. The robot's RIGHT is turned ON.

In practice, there exists a short delay between the moment the robot senses that all conditions for standstill are satisfied and the moment the robot effectively comes to a standstill. We take this delay into account by introducing a small real constant ϵ such that each Run-Mode robot comes to a Standstill at a distance from the obstacle with value in $(d, d + \epsilon)$.

To make sure subgroups of robots remain connected, we add two more conditions to change from Run-Mode to Standstill:

- Switch to Standstill if your IF is at Standstill and its LEFT is ON.
- Switch to Standstill if your IL is at Standstill and its RIGHT is ON.

In rare cases it is possible that a robot switches to standstill and turns both the LEFT and RIGHT parameter ON. Figure 8 presents such a situation considering a

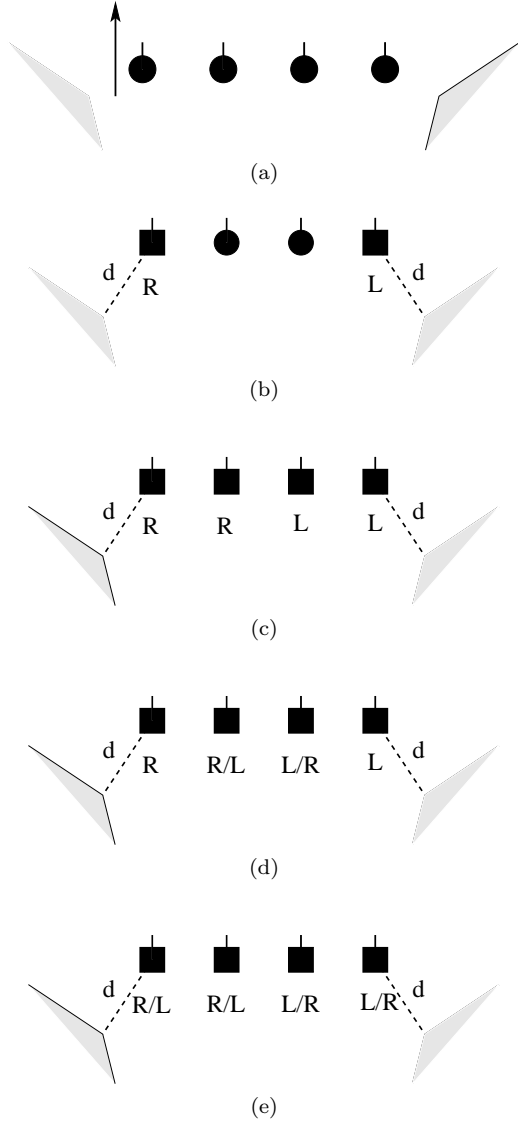


Figure 8: A group of four Run-Mode robots moving between two obstacles and turning to Standstill with each robot activating both LEFT and RIGHT parameter.

robot subgroup of four robots. The group is able to move through the gap created by two obstacles (partially shown in the figure) (a). Because of the symmetry of the configuration, the outer robots come to a standstill at the same time. The leftmost/rightmost robot turns RIGHT/LEFT ON (b). Shortly after, the two inner robots switch to Standstill. The second robot turns RIGHT ON, the third activates LEFT (c). In Algorithm 3 treating Standstill behavior it is shown that the activation of a RIGHT or LEFT parameter propagates through an entire group of Standstill robots (d). In the situation under consideration this leads to all four robots having both LEFT and RIGHT parameters turned ON (e).

5.3. Switching from Contour-Following to Run-Mode or Standstill

The above rules lead to situations with one or more robot groups at Standstill and other robots Contour-Following around the obstacle. The key idea of the algorithm is to make Contour-Following robots (re)join subgroups of Standstill robots, in order to advance further inside the strip. Figure 7 shows the three possible situations for subgroups of Standstill robots. (Robots indicated by a square are at Standstill; robots depicted by a diamond are Contour-Following.) A group of Standstill robots can be located

1. on the left of an obstacle (Situation 1) ,
2. at a top of an obstacle (Situation 2) ,
3. on the right of an obstacle (Situation 3).

A Contour-Following robot encounters a group of Standstill Robots by detecting a Standstill robot with an available RNP or LNP. The robot then moves to this location and occupies it.

5.3.1. Situation 3

If the position now occupied is the LNP of a Standstill robot, a subgroup of Standstill robots in Situation 3 has been reached. In general, each of these Standstill robots has its RIGHT parameter turned ON. However, the Contour-Following robot first checks the LEFT parameter of its new neighbor, in order to detect rare cases like the situation discussed in Figure 8. If LEFT is ON, this means that the group of Standstill robots under consideration is not only located on the right side of an obstacle (namely the obstacle the Contour-Following robot was moving around) but also on the left side of another obstacle. The Contour-Following robot takes on the Standstill mode and turns its LEFT parameter ON; its RIGHT parameter remains OFF. This will cause a turning off of the RIGHT parameter in the entire group of Standstill robots (see Algorithm 3), resulting in a Standstill group in Situation 1 or 2.

On the other hand, if the LEFT parameter of the new neighbor is OFF, the Contour-Following robot simply turns to Run-Mode. The Standstill robots respond by turning RIGHT off, resulting in a switch to Run-Mode of the entire Standstill group (see Algorithm 3).

Often the Contour-Following robot and the leftmost robot of the Standstill group are not an original leader-follower pair. This implies that they have to actively construct a new leader-follower connection between each other so that they can proceed inside the strip according to the Run-Mode part of the algorithm (see Algorithm 2).

5.3.2. Situations 1 and 2

In case the Contour-Following robot occupies the RNP of a Standstill robot, the robot has to determine whether it reached a group in Situation 1 or Situation 2. If it is concluded that a group in Situation 1 has been reached

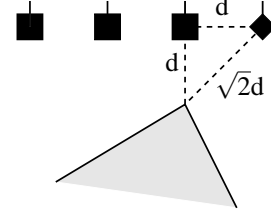


Figure 9: A Contour-Following robot (diamond) has reached the RNP of a Standstill robot (square) at the top of an obstacle. The figure depicts the largest possible distance between the Contour-Following robot and the obstacle.

Algorithm 2 TYPE Contour-Following

```

Repeat
  Move clockwise around obstacle,
until
  if A robot of Contour-Following or Stand-Still type
  blocks your path then
5:   Wait
  else if Moving upward inside the strip  $\wedge$  RNP(IL) is
  reachable  $\wedge$  LEFT(IL) = ON  $\wedge$  TOP(IL) = OFF then
    Move to RNP(IL)
    if RIGHT(IL)=ON then
      RIGHT  $\leftarrow$  ON
10:   TYPE  $\leftarrow$  Standstill
    else
      TYPE  $\leftarrow$  Run-Mode
    end if
  else if Moving upward inside the strip  $\wedge$  RNP(IL) is
  reachable  $\wedge$  LEFT(IL) = ON  $\wedge$  TOP(IL) = ON then
15:   Keep on Contour-Following
  else if Moving downward inside strip  $\wedge$  you meet a
  robot with RIGHT=ON, TYPE Standstill and LNP
  reachable then
    Move to LNP of this robot
    Assign the new robot as your IF
    Assign yourself as the new robot's IL
20:   if LEFT(IF)=ON then
    LEFT  $\leftarrow$  ON
    TYPE  $\leftarrow$  Standstill
    else
      TYPE  $\leftarrow$  Run-Mode
25:   end if
  else if Moving downward inside strip  $\wedge$  you meet a
  robot with TYPE Run-Mode then
    Stay at a fixed minimal distance from this robot
    and retrace your steps along obstacle if necessary
  end if

```

then a procedure similar to the one for situation 3 is performed. To obtain this procedure, just switch “LEFT” and “RIGHT” in the description of Section 5.3.1. However, if Situation 2 is detected, the Contour-Following robot will abandon the RNP location to continue around the obstacle in search of a Standstill group in Situation 3. The Standstill robot at the top of the obstacle will turn its TOP parameter ON, ensuring other Contour-Following robots encountering the robot pass it without checking the Standstill group.

The distinction between Situation 1 and 2 is easy to detect. The Contour-Following robot, located at the RNP of its IL, checks the direction along which it detects the obstacle it was moving around. If this direction belongs to the fourth quadrant of the Contour-Following robot's surroundings, Situation 1 holds; if it belongs to the third quadrant, Situation 2 is detected.

Remark that we do not allow the robot to record data on the environment. A robot is not able to remember which obstacle it was moving around a time instant earlier. However, if a Contour-Following robot joins a standstill group in situation 1 or 2 this information is required, as explained above. Therefore we introduce the assumption which leads to a minimum inter-obstacle distance.

Assumption 1. *When a Contour-Following robot joins a Standstill group, the obstacle the robot was moving around is the nearest obstacle of all obstacles detected by the robot, i.e. the obstacle belonging to the robot's measured Dist2O.*

The largest possible distance between a robot that just joined a standstill group and the obstacle it was moving around is depicted in Figure 9. Its size is computed to be $\sqrt{2}d$. It follows from Assumption 1 that all other obstacles are located further away from the robot. This imposes a lower bound on the inter-obstacle distances: the minimum allowable interdistance equals $2\sqrt{2}d$.

5.4. Switching from Standstill to Run-Mode

The previous sections showed that whenever a robot switches to Standstill, it also turns ON one of the parameters LEFT and RIGHT. These parameters determine the

behavior of the Contour-Following robot joining a Standstill robot.

A Standstill robot will switch back to Run-Mode if both its LEFT and RIGHT parameters are turned OFF. Each Standstill robot copies the behavior of its left neighbor to turn the RIGHT parameter on or off, and similarly copies the behavior of its right neighbor to turn LEFT on or off.

Algorithm 3 TYPE Standstill

```

Repeat
  Stand still and
  if your IL is at your LNP  $\wedge$  RIGHT(IL) = ON/OFF
  then
    RIGHT  $\leftarrow$  ON/OFF
5: else if your IF is at your RNP  $\wedge$  LEFT(IF) = ON/OFF
  then
    LEFT  $\leftarrow$  ON/OFF
  end if
until
if LEFT = OFF  $\wedge$  RIGHT = OFF then
10: TYPE  $\leftarrow$  Run-Mode
  else if LEFT = ON  $\wedge$  your IL is at your LNP  $\wedge$  a new
  robot in Run-Mode appears at RNP then
    Assign the new robot as your IF
    Assign yourself as the new robot's IL
     $\triangleright$  (comment) This reconnects two subgroups at
    the top of an obstacle
15: end if

```

5.5. Obstacles on the strip boundary

As mentioned in the introduction, obstacles are allowed to be located on a strip boundary. The algorithm will treat such obstacles by sending (part of) the robot group around the obstacle *outside of the strip*. To execute this maneuver, the algorithm of the Interior Robots does not need to be changed. Both LSB and RSB are given an algorithm which slightly differs from the other robots. When its forward path is obstructed by an obstacle, the LSB (RSB) robot moves clockwise (counterclockwise) around the obstacle *outside of the strip*, until it reaches its original horizontal position. This location is called the *reentry point* and is indicated by a letter X in Figure 10. In order to measure this position the robot is equipped with GPS. Once the reentry point has been reached, one of four possible configurations is detected, as depicted in Figure 10, each requiring a different treatment.

First, consider Figure 10a. The obstacle will cause the robot group to split between the third and the fourth robot. The LSB moves clockwise around the obstacle and reaches its reentry point. Resuming its basic algorithm, it moves parallel to the strip boundary until the distance between the obstacle and itself exceeds the preset threshold d . This creates a situation similar to situations 1 or 2 of Figure 7. Interior Robots 2 and 3, moving clockwise

around the obstacle, will reconnect to the LSB or, if the LSB is located at the top of the obstacle, will continue along the obstacle to expand the robot subgroup located on the right side of the obstacle. Without further adjustments the basic algorithm is able to tackle this situation as required.

In this respect, the configuration in Figure 10b causes similar behavior. All robots move along the left of the obstacle, except for the RSB. The RSB reaches its reentry point after which it moves forward up to the preset distance d with the obstacle. The situation attained is similar to situation 3 of Figure 7. Again, no further adjustments are needed to let the basic scanning algorithm finish the job satisfactorily.

The configurations depicted in Figure 10c and 10d do *not* satisfy the pattern observed in the previous two cases. When the RSB reaches the reentry point, a previously unencountered situation is created. The slope of the obstacle's boundary near the reentry point does not allow any of the situations 1, 2, or 3. The algorithm of the RSB needs to be extended. After reaching its reentry point, we let the RSB sense the orientation of the slope of the obstacle boundary. The robot does this by checking the direction along which it detects the obstacle under consideration. If this direction belongs to the fourth quadrant of the RSB, as depicted in Figure 10c, the robot abandons its standard behavior and remains positioned at the reentry point instead. It waits for its IL to appear within its sensor range. This IL will end up as the rightmost robot of a Standstill robot group in situation 1. The RSB is able to detect this and can reoccupy the RNP of its IL. The RSB completes the robot group and switches back to its standard behavior.

Finally, the situation in (Figure 10d) is addressed. Similar to the previous case, after sensing the obstacle boundary slope at the reentry point, the LSB switches to an extension of its original program. It moves forward until the distance between the obstacle and itself exceeds the preset threshold, at which point it waits until all robots which followed it clockwise around the obstacle have passed. These robots expand the robot group on the right side of the obstacle. The LSB retraces its steps towards the obstacle until it reaches the LNP of the leftmost robot of the (right) robot group. The LSB connects to this robot completing the robot group. The obstacle has been tackled successfully and the sweeping of the strip can continue.

5.6. Simulation results

The algorithm presented in this section has been successfully implemented in Matlab code. Our program treats diverse obstacle configurations. Figure 6 is the result of one such run of the program where a situation with two obstacles is considered. The figure displays 8 snapshots taken during the simulation, with reading order left to right, top to bottom. To better verify the capabilities of the program, the output of some runs of the program have been recorded in the form of videos. These videos can

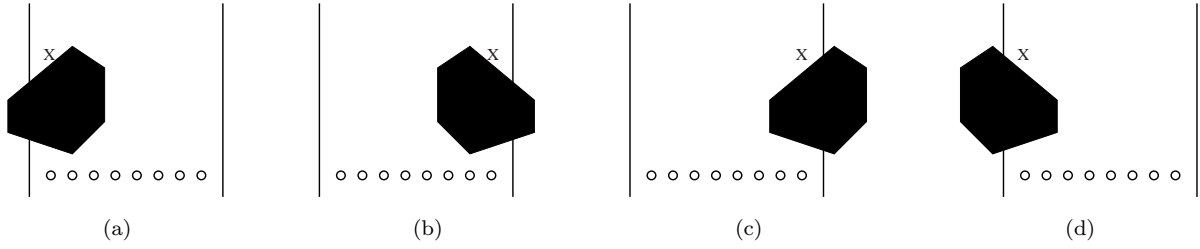


Figure 10: Four distinct cases of an obstacle located on the strip boundary. The two situations on the right demand an extension of the algorithm, the other two do not.

be retrieved at <http://www.systems.ugent.be/videos/>. In these animated simulations a color code has been used to indicate the status of each robot, instead of different shapes of the symbols representing the robots.

The simulations allow us to quantify the time gain obtained from using a robot team compared to single-robot coverage. The single-robot coverage strategy we take into consideration for comparison is the algorithm executed by the LSB robot in the present paper. In a single-robot setting, this robot sweeps a strip of one robot wide using GPS, and whenever it encounters an obstacle, it moves clockwise around it until it reaches its assigned strip again.

We conducted simulations for both a group of 10 robots and a single robot. In case of an empty strip the 10-robot group sweeps the strip 10 times faster than the single robot, as expected. In the presence of obstacles the time gain is reduced: our simulations show that the group of 10 robots finishes a strip 6 – 8 times faster than the single robot. The exact value of the time gain depends on the obstacle configuration under consideration.

6. Proof of coverage

In Section 4 it was shown how the surface A of a strip was covered when the robot team tracks a set V of parallel lines with interdistance d . In the presence of obstacles, this proposition holds with V and A replaced by $V \setminus \mathcal{P}$ and $A \setminus \mathcal{P}$ respectively. We add the following definition.

Definition 7. A set consisting of all points with the same x -value belonging to $V \setminus \mathcal{P}$ is called a basic robot track. Each connected subset of a basic robot track is called a section of a basic robot track.

We conclude that, in order to prove coverage of the entire area, it is sufficient to prove coverage of all basic robot track sections. This is the subject of the present section. For simplicity we restrict to situations where

- the basic robot tracks of the LSB and RSB are not obstructed by obstacles,
- all obstacles are convex with exactly one top.

Lemma 1. Each robot in Run-Mode moves along a trajectory with constant x -coordinate.

PROOF. The robot group under consideration is finite, which implies the existence of one or more disjoint finite subgroups of Run-Mode robots in a straight-line formation and with consecutive indices. If a robot is the leftmost robot of a Run-Mode subgroup, it satisfies at least one of the following conditions: the robot

- has no IL,
- does not sense its IL, or
- senses its IL, which is Contour-Following.

This is proven by contrapositive: assume that for a robot none of the above three conditions is satisfied. Then one of the following situations holds:

1. The robot senses its IL which is at Standstill. Consequently the robot turns to Standstill, and hence is not in Run-Mode.
2. The robot senses its IL which is in Run-Mode. Consequently the robot is also in Run-Mode but is not the leftmost robot of a Run-mode subgroup.

From the three conditions above and the definition of Run-Mode (see Section 3.3), it follows that the leftmost robot of a Run-Mode subgroup moves with constant x -coordinate. The remaining robots of the Run-Mode subgroup copy this movement, since they all sense their respective IL in Run-Mode.

Lemma 2. Consider a scanning strip with boundaries at $x = x_L$ and $x = x_R$, with $x_L < x_R$. Let P be an obstacle with x_{min} the x -coordinate of its leftmost point(s) and x_{max} the x -coordinate of its rightmost point(s). Eventually all robots moving along a basic robot track section with the x -coordinate belonging to $[\max(x_L, x_{min}), \min(x_R, x_{max})]$ and the end-point at P , will switch to Contour-Following around P .

PROOF. The only situation where a robot with the properties described above will *not* start Contour-Following around P is when it switches from Run-Mode to Standstill before reaching the obstacle. We investigate if this situation can occur. Standstill can only be caused by the presence of an obstacle Q_i (different from P because of the convexity assumption). The Stand-till robot belongs

to a subgroup in one of the situations 1, 2 or 3 presented in Section 5.3, related to an obstacle Q_i characterized by leftmost and rightmost points ($x_{i,min}$ and $x_{i,max}$). The robot remains at Stand-still if the corresponding Standstill group does not get joined by a robot Contour-Following around Q_i . It is easy to see that this implies that not all robots with coordinate $x \in [\max(x_L, x_{i,min}), \min(x_R, x_{i,max})]$ start Contour-Following around Q_i (or that at least one Contour-following robot gets obstructed by a persistent Standstill group belonging to an obstacle $Q_j \neq Q_i$).

This observation shows that the original assumption at obstacle P (i.e. not all robots with x-coordinate in $[\max(x_L, x_{min}), \min(x_R, x_{max})]$ will start contour-Following around P) leads to the same situation at an obstacle Q_i different from P . Consequently, the initial assumption implies the existence of an infinite series of different obstacles. Since the number of obstacles is assumed to be finite, the original assumption is not valid. This concludes the proof.

We now present the proof of coverage.

Theorem 1. *If the minimum inter-obstacle distance is $2\sqrt{2}d$, then the algorithm covers all basic robot tracks completely.*

PROOF. At the initialization of the algorithm, the robot group has assumed the formation defined in Section 2.3. Each robot is located at the start of a basic robot track. It follows from Lemma 1, that at the beginning of the algorithm each robot moves in Run-Mode along its basic robot track. A robot abandons the basic robot track it was moving on, if and only if it has switched to Contour-Following. We have to prove that

- every Contour-Following robot switches back to Run-Mode if and only if it is located on a basic robot track,
- each section of a basic robot track is traced once,

Consider an obstacle P characterized by x_{min} and x_{max} . The robots moving along basic robot tracks with $x \in [x_{min}, x_{max}]$ will reach P and switch to Contour-Following (cfr. Lemma 2). The two robots located on the basic robot tracks with $x \in [x_{min} - d, x_{min})$ or $x \in (x_{max}, x_{max} + d]$ move past the obstacle. Call these robots R_L and R_R respectively. (These robots exist, due to the assumption of unobstructed robot tracks for LSB and RSB.) These two robots come to a standstill, when the distance between itself and obstacle P satisfies the following conditions (cfr. Section 5.2):

- has a value in $(d, d + \epsilon)$, $\epsilon \ll 1$,
- it is the shortest of all distances between the robot and all obstacles surrounding it.

Satisfaction of the second condition is guaranteed by the theorem's assumption on the minimum inter-obstacle distance. At standstill, both robots R_L and R_R lack a neighboring robot on the side where the obstacle is located.

Since the robots are located more than d distance units away from any obstacle they have either an unoccupied LNP (in the case of R_R) or unoccupied RNP (for R_L) that is not obstructed by an obstacle. The robots R_L and R_R are located on basic robot tracks, so their LNP and RNP are located on basic robot tracks as well. Since P has exactly one top, one of the following configurations has been attained:

1. LNP of R_R at the top, RNP of R_L belonging to situation 3 (as defined in Section 5.3);
2. LNP of R_R belonging to situation 1, RNP of R_L at the top;
3. LNP of R_R belonging to situation 1, RNP of R_L belonging to situation 3;

In each of the three cases, a Contour-Following robot has the opportunity to occupy either a free LNP or a free RNP. From Lemma 2 and the description of the algorithm in Section 5 it follows that the number of Contour-Following robots available is necessary and sufficient to occupy the free RNPs/LNPs. The theorem's assumption ensures that the Contour-Following robots are able to tackle the top of the obstacle which guarantees that every available LNP will be reached by a Contour-Following robot, as explained in Section 5.3. When a Contour-Following robot has obtained the free RNP/LNP, its sensors are covering the part of its basic robot track that lies between itself and the obstacle it was moving around. The robot turns to Run-Mode and starts tracing the rest of the respective basic robot track section. The entire reasoning followed for R_L or R_R can now be repeated for the Contour-Following robot which has switched back to Run-Mode. Every time a Contour-Following robot comes to Standstill it creates an available RNP or LNP to be occupied by a next Contour-Following robot, until the two robot groups merge at the top of the obstacle. It follows that each section of a basic robot track is traced by precisely one robot.

7. Conclusion

This paper describes an algorithm for multi-robot coverage in an unknown environment. The algorithm uses robot formations, which

- reduces the need for extensive sensor capabilities,
- keeps radio communication between robots at a minimum,
- enables us to treat both coverage and pursuit-evasion missions.

The robots scan the environment along predefined strips. Information on location of the strips is loaded into the memory of both outer robots LSB and RSB at the beginning of the algorithm and remains unchanged. Hence, a cellular decomposition of the environment that is dynamically created during the algorithm, is not called for.

Such a decomposition requires more memory capacity of the robots and more inter-robot communication. Dividing the area into fixed strips requires an intelligent algorithm to pass the obstacles located in the environment. Our algorithm successfully treats all possible configurations of convex obstacles. As demonstrated in simulations, the robot team is able to pass multiple obstacles simultaneously as well as obstacles located on the strip boundary. A proof of coverage concludes the paper.

Acknowledgment

This paper presents research results of the Belgian Network DYSCO (Dynamical Systems, Control, and Optimization), funded by the Interuniversity Attraction Poles Programme, initiated by the Belgian State, Science Policy Office. The scientific responsibility rests with its authors.

References

- [1] J. Ota, Multi-agent robot systems as distributed autonomous systems, *Advanced engineering informatics* 20 (2006) 59 – 70.
- [2] D. Keymeulen, J. Decuyper, The fluid dynamics applied to mobile robot motion: the stream field method, in: *Proceedings of 1994 IEEE International Conference on Robotics and Automation*, Piscataway, NJ, USA, pp. 378–385.
- [3] D. Spears, W. Kerr, W. Spears, Physics-based robot swarms for coverage problems, *International Journal Intelligent Control Systems* 11 (2006) 124–140.
- [4] W. Burgard, M. Moors, C. Stachniss, F. Schneider, Coordinated multi-robot exploration, *IEEE Transactions on Robotics* 21 (2005) 376–386.
- [5] N. Agmon, N. Hazon, G. A. Kaminka, Constructing spanning trees for efficient multi-robot coverage, in: *Proceedings of the 2006 IEEE International Conference on robotics and Automation*, volume 1–10, Orlando, FL, USA, pp. 1698–1703.
- [6] N. Hazon, F. Miele, G. A. Kaminka, Towards robust on-line multi-robot coverage, in: *Proceedings of the 2006 IEEE International Conference on robotics and Automation*, volume 1–10, Orlando, FL, USA, pp. 1710–1715.
- [7] X. Zheng, S. Jain, S. Koenig, D. Kempe, Multi-robot forest coverage, in: *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, pp. 2318–2323.
- [8] Z. Butler, A. Rizzi, R. Hollis, Distributed coverage of rectilinear environments, in: *Proc. of the Workshop on the Algorithmic Foundations of Robotics*, pp. 51–61.
- [9] C. Luo, S. Yang, D. Stacey, Real-time path planning with deadlock avoidance of multiple cleaning robots, in: *Proceedings 2003 IEEE International Conference on Robotics and Automation*, volume 3, pp. 4080 – 4085.
- [10] S. Koenig, B. K. Szymanski, Y. Liu, Efficient and inefficient ant coverage methods, *Annals of Mathematics and Artificial Intelligence* 31 (2001) 41–76.
- [11] I. A. Wagner, Y. Altshuler, V. Yanovski, A. M. Bruckstein, Co-operative cleaners: A study in ant robotics, *The International Journal of Robotics Research* 27 (2008) 127–151.
- [12] R. Menezes, F. Martins, F. E. Vieira, R. Silva, M. Braga, A model for terrain coverage inspired by ant’s alarm pheromones, in: *Proceedings of the 2007 ACM symposium on Applied computing (SAC07)*, New York, pp. 728–732.
- [13] G. Kowadlo, R. A. Russell, Robot odor localization: A taxonomy and survey, *International Journal of Robotics Research* 27 (2008) 869–894.
- [14] C. S. Kong, N. A. Peng, I. Rekleitis, Distributed coverage with multi-robot system, in: *Proceedings of 2006 IEEE International Conference on Robotics and Automation*, Orlando, Florida, USA, pp. 2423–2429.
- [15] D. Latimer-IV, S. Srinivasa, V. Lee-Shue, S. S. Sonne, H. Choset, Toward sensor based coverage with robot teams, in: *Proc. 2002 IEEE International Conference on Robotics and Automation*, pp. 961–967.
- [16] I. M. Rekleitis, A. P. New, H. Choset, Distributed coverage of unknown/unstructured environments by mobile sensor networks, in: *3rd International NRL Workshop on Multi-Robot Systems*, Washington, D.C., pp. 145–155.
- [17] I. Rekleitis, A. P. New, E. S. Rankin, H. Choset, Efficient boustrophedon multi-robot coverage: an algorithmic approach, *Annals of Mathematics and Artificial Intelligence* 52 (2008) 109–142.
- [18] E. Garcia, P. G. de Santos, Mobile robot navigation with complete coverage of unstructured environments, *Robotics and Autonomous Systems* 46 (2004) 195–204.
- [19] J. Rogge, D. Aeyels, Multi-robot coverage to locate fixed and moving targets, in: *Proceedings of the 3rd IEEE Multi-conference on systems and control*, pp. 902–907.
- [20] D. Gage, Many-robots MCM search systems, in: *Proceedings of Autonomous Vehicles in Mine Countermeasures Symposium*, pp. 956–964.
- [21] S. I. Sundaraja, R. R. Brooks, *Distributed sensor networks*, CRC Press, 2004.
- [22] A. Elfes, Using occupancy grids for mobile robot perception and navigation, *Computer* 22 (1989) 46–57.